BISM-II Multipoint Firmware

User's Guide
Version 1.2

global solutions: local support™

Americas: +1-800-492-2320 Option 2
Europe: +44-1628-858-940
Hong Kong: +852-2923-0610

REVISION HISTORY

Revision	Date	Description
1.0	23 May 2012	Initial version
1.1	14 August 2013	Updated PinCode parameters in section 3.5.3 PinCode
1.2	31 October 2013	Updated Max. Slave connections

CONTENTS

Revision History 1
1 Introduction 5
1.1 Flow control & Data Integrity 5
1.2 Maximum Packet Size Consideration 5
2 Packet General Format 6
2.1 Host to Module Packets 6
2.1.1 Command Packets 6
2.1.2 Data Packets 7
2.2 Module to Host Packets 7
2.2.1 Response Packets 7
2.2.2 Event Packets 8
2.2.3 Data Packets 8
2.3 Host Packet Receive Flowchart 9
3 Host Command/Responses 10
3.1 Information Commands 10
3.1.1 No Operation 10
3.1.2 Get Connectable, Discoverable, Security Modes 11
3.1.3 Read Local Bluetooth Address 11
3.1.4 Information 12
3.2 Configuration Commands 13
3.2.1 Read 'S' Register 13
3.2.2 Write 'S' Register 14
3.2.3 Store 'S' Registers 15
3.2.4 Default 'S' Registers 15
3.2.5 Read Communications Parameters 16
3.2.6 Write Communications Parameters 16
3.3 Connection Commands 17
3.3.1 Set Connectable Mode 17
3.3.2 Service Incoming Connection 18
3.3.3 Make Outgoing Connection 19
3.3.4 Drop Connection 20
3.3.5 Set Modem Lines 20
3.3.6 SCO Connect 21
3.3.7 SCO Disconnect 22
3.3.8 SCO Incoming Setup 23
3.3.9 Sniff Request 24
3.3.10 Park Request 25
3.3.11 RSSI and Link Quality 26
3.3.12 Stream Connect 27
3.3.13 Stream Disconnect 28
3.4 Inquiry Commands 28
3.4.1 Inquiry Request 28
3.4.2 Set Discoverable Mode 29
3.5 Pairing Commands 30
3.5.1 Pair Initiate 30
3.5.2 Pair Accept 31
3.5.3 PinCode 32
3.5.4 Trusted Database Record Count 33
3.5.5 Trusted Database Read Record 33
3.5.6 Trusted Database Delete Record 34
3.5.7 Trusted Database Change Type 35

3.5.8 Trusted Database Is Peer Trusted 35
3.6 Miscellaneous Commands 36
3.6.1 Set/Get Security Mode 36
3.6.2 Get Remote Friendly Name 36
3.6.3 Get Local Friendly Name 38
3.6.4 Set Local Friendly Name 38
3.6.5 Get Digital/Analog I/O 39
3.6.6 Set Digital I/O 40
4 Module Events 41
4.1 Inquiry Events 41
4.1.1 Inquiry Result 41
4.2 Information Events 41
4.2.1 Unknown Command 41
4.2.2 Status 42
4.2.3 Invalid Packet Size 42
4.3 Connection Events 43
4.3.1 Connection Setup 43
4.3.2 Incoming Connection 43
4.3.3 Disconnect 44
4.3.4 Modem Status 44
4.3.5 Low Power Mode 45
4.3.6 SCO Connect 45
4.3.7 SCO Disconnect 46
4.3.8 SCO Incoming Setup 46
4.3.9 Remote Features 46
4.4 Miscellaneous Events 47
4.4.1 Link Key 47
4.4.2 Pin Code Request 48
4.4.3 Local Friendly Name 48
4.4.4 Remote Friendly Name 48
4.4.5 ADC value 49
4.5 Debug Events 49
4.5.1 Debug Packet 49
5 Miscellaneous 50
5.1 S Registers 50
5.2 COMMAND & EVENT values 52
5.3 STATUS values 52

1 INTRODUCTION

This document describes a packet based messaging interface which is used by a host to send commands, receive responses and exchange multiplexed data with the **blu2i Multipoint Module**, henceforth described as the Module.

The Module consists of the same hardware as that used for the single point 'AT' blu2i module described elsewhere. Whereas the latter only allows one-to-one connection, the module described here allows simultaneous connections to a maximum of 3 slaves. It also allows connections to multiple profiles to one or more slaves. Hence this document adopts a concept of channels instead of slave connections.

The term 'host' in this document is taken to mean any entity which is a source of command messages, sink for response/event messages and both source and sink for multiplexed data packets.

To further eliminate any confusion, when the term 'command message' is used, it implies a message from the host to the module and likewise 'response message' is used to imply a message from the module to the host.

This document does NOT describe how the packets are physically exchanged between the host and the module. The transport medium will be either UART or USB. It also does NOT describe the format of any envelope that may be required to reliably and quickly transfer the message packet between the host and module.

This implies that when the packets proposed in this document are processed, they are assumed not to contain any errors.

1.1 Flow control & Data Integrity

It must be recognised that the transport mechanism will be streaming in nature. If the transport medium is USB then flow control and data integrity is inherently provided by the USB protocol.

If UART is the medium, then it shall be assumed that there will be a 5 wire interface, RX,TX,CTS,RTS and GND. Any host attached to the UART of the module shall **strictly** observe CTS/RTS hardware handshaking. Packet data integrity may or may not be provided depending on the build. It is expected that for a UART transport media, guaranteeing data integrity will be at the **severe** expense of data throughput.

1.2 Maximum Packet Size Consideration

Ezurio acknowledges that hosts attaching to the module may have limited RAM (random access memory) resource and so the module makes provision for this by guaranteeing not to send packets larger than a certain negotiated size (See S Reg 2, in Chapter 5). However, it is assumed that there will be a certain minimum packet size (16) that the host SHALL be capable of coping with. The host must guarantee that there is space for at least 2 packet in the receive buffer.

2 PACKET GENERAL FORMAT

This section describes the general format of incoming and outgoing packets. The term 'incoming' will henceforth imply packets sent by the host to the module and 'outgoing' in the reverse direction. That is, the direction terminology is module centric.

All packets have octet granularity. When an octet is described as containing bit fields, it shall be taken that bit 0 is the least significant bit and bit 7 is the most significant bit.

Subfields in the packet which require multiple octets, shall be ordered so that the lowest significant octet is transmitted LAST over the transport media, unless specifically described otherwise. For example, a 16 bit word value will require 2 octets within the packet and the first octet will correspond to the upper byte. Similarly, a 6 byte Bluetooth address shall be transported most significant byte first. If the order is reversed then it will be specifically highlighted in the description of appropriate packets.

Subfields which are data arrays shall be described with the '[ ]' operator in descriptions which come in subsequent chapters.

Apart from data packets, all command, respond and event packets are of fixed size. If there is enough data to fill a packet, then the packet is filled with 0s.

The protocol and packet format is optimised to ensure maximum data throughput via the virtual machine which processes these packets, and as such shall allow a maximum of 3 channels. It is possible to have all 3 channels in a single slave connection (to multiple profiles).

In recognition of the fact that a host may have limited RAM, the command/response will be designed to minimise the size of response packets. This means, the number of commands may seem unduly large where multiple command packets will be required to get a set of data, where it would logically have been prudent to send all the data in one large response packet.

The design aims to optimise throughput over the air. The following sections describe the packets in detail.

2.1 Host to Module Packets

These are packets used to convey commands to the module or raw data to be sent over an open Bluetooth RFCOMM connection.

2.1.1 Command Packets

The format for command packets is as per the table below.

Octet	Field	Description
0	LENGTH	Total length of this packet, including this octet
1	CHANNEL	Always 0
2	COMMAND	Described in the next chapter
3	FLOW_IN	Bit 0 to 6 specify a mask. A clear bit means the module should NOT send any more packets to that corresponding data channel. Bit 7 is always 0 and will be used as an extension bit in the future. It is assumed that the host will always be able to receive a response or status packet.
4..N	DATA[]	Data as required and has meaning specific to COMMAND. For example, if the command is to make a connection to a peer device, then it will be at least a 6 octet array specifying the Bluetooth address of the peer.

The value of COMMAND shall be in the range 0 to 127.

Unknown command values result in an EVT_UNKNOWN_COMMAND event, with the command value reflected in the data field. If the octet value is specified in the range 128 to 255 (0x80 to 0xFF), then reflecting that value in the data field of an EVT_UNKNOWN_COMMAND instead of the COMMAND field of a response packet guarantees that the packet will NOT be mistakenly processed as an event.

2.1.2 Data Packets

The format for data packets is as per the table below and can arrive at any time. The only method by which the host can be stopped from sending this message is by sending a 0 value in the FLOW_OUT field of a response or status message. The module will be prepared to receive at least one data packet after deasserting the appropriate flow control bit.

If CHANNEL is 0xFF, then the data packet is broadcast to all open channels. (Future feature).

Octet	Field	Description
0	LENGTH	Total length of this packet, including this octet
1	CHANNEL	Bits 0 to 4 contain the value as per the channel id value returned in a successful connection message. It will be a value in the range 1 to 31. Bits 5,6,7 are reserved for future use and should always be set to 0.
2..N	DATA[]	This data array is unconditionally sent over the air

2.2 Module to Host Packets

These are packets used to convey responses or events from the module and raw data received over an open Bluetooth RFCOMM connection. Response packets **shall** always be as a result of a command packet and event packets are **asynchronously** sent to the host as and when required. *The host shall ensure that it is always ready to accept response and event packets, especially event packets as they can be sent at any time.*

2.2.1 Response Packets

The format for response packets is as per the table below.

Octet	Field	Description
0	LENGTH	Total length of this packet, including this octet
1	CHANNEL	**Always 0**
2	COMMAND	Echoed from the command packet (Shall be > 0)
3	FLOW_OUT	Bit 0 to 6 specify a mask. A clear bit means the host should NOT send any more packets to that corresponding data channel. Bit 7 is always 0 and will be used as an extension bit in the future.
4	STATUS	0 means success
N..M	DATA[]	Data as required and has meaning specific to the response for COMMAND

2.2.2 Event Packets

The format for status packets is as per the table below.

Octet	Field	Description
0	LENGTH	Total length of this packet, including this octet
1	CHANNEL	Always 0
2	EVENT	Described later
3	FLOW_OUT	Bit 0 to 6 specify a mask. A clear bit means the host should NOT send any more packets to that corresponding data channel. Bit 7 is always 0 and will be used as an extension bit in the future.
N..M	DATA[]	Data as required and has meaning specific to the response for COMMAND

The only difference between a response and an event packet is that in the latter, octet 2 is defined as COMMAND in the former and EVENT in the latter. Also in the latter, the STATUS field is missing.

The value of COMMAND shall be in the range 0 to 0x7F and EVENT shall take values in the range 0x80 to 0xFF. This allows bit 7 of that octet to be used to decode whether the packet is a response packet or an event packet.

The value of STATUS is in the range 0 to 255. A value of 0 means SUCCESS and any other value is a failure, where the value gives more details of the failure type. The values of STATUS are defined in a 'C' header file which can be obtained on request from Ezurio.

2.2.3 Data Packets

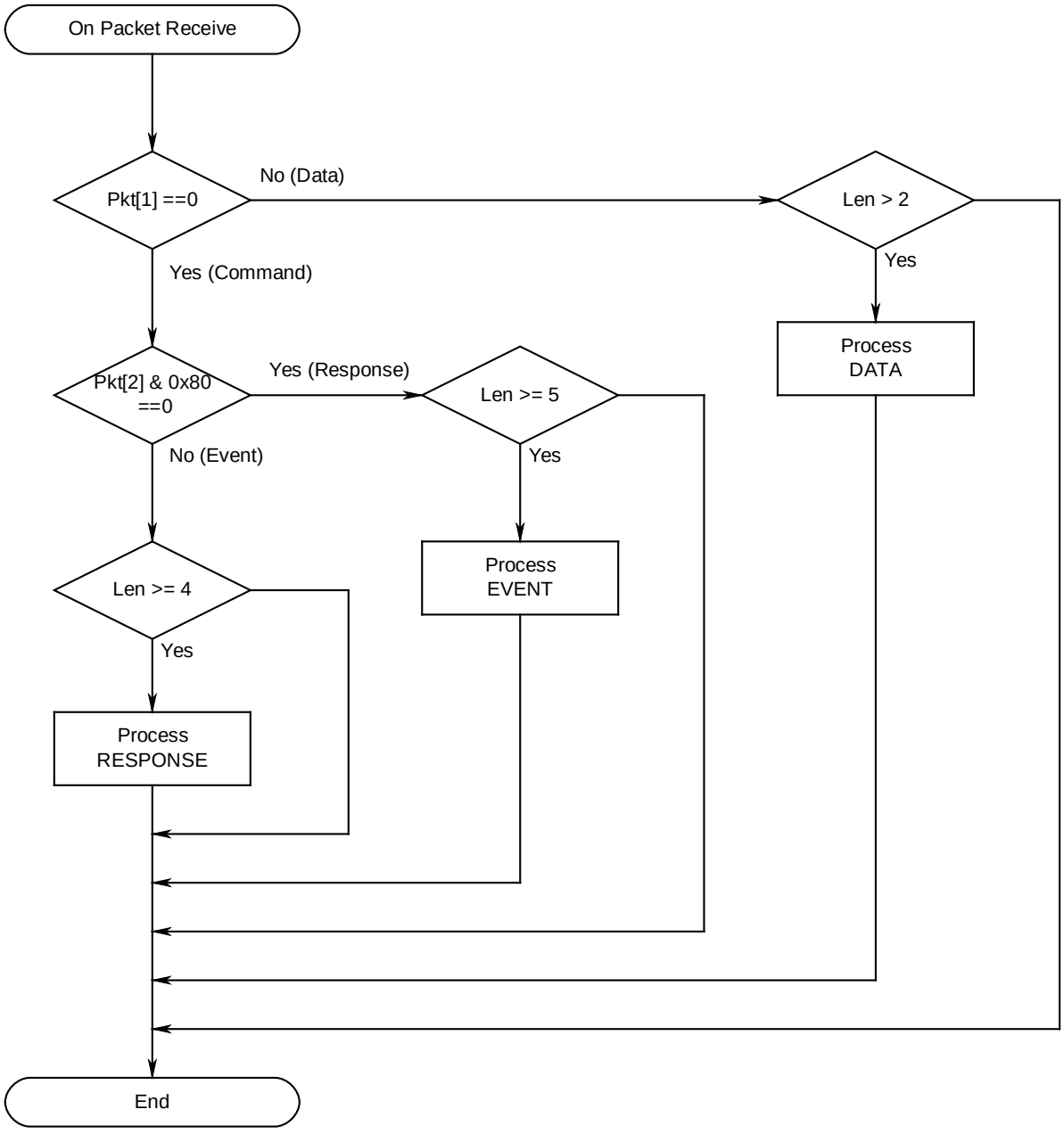
The format for data packets is as per the table below. The only method by which the host can stop the module from sending this message is by sending a 0 value in the FLOW_IN field of command message.

Octet	Field	Description
0	LENGTH	Total length of this packet, including this octet
1	CHANNEL	Bits 0 to 4 contain the value as per the channel id value returned in a successful connection message. It will be a value in the range 1 to 31. Bits 5,6,7 are reserved for future use and should always be set to 0.
2..N	DATA[]	Data received over the air for the channel

Data packets are symmetrical in both directions.

2.3 Host Packet Receive Flowchart

As maximum data throughput is the design goal, the format and detail of packets have been optimised. Implement the following flowchart, in the host, for rapid servicing and flow control of the packets.



3 HOST COMMAND/RESPONSES

This section describes all host commands in detail and is specified via the COMMAND field of all command packets. The description for each command below is in the form of a command packet table and a corresponding response packet table.

Each command has a unique COMMAND value in the range 1 to 127 (0x01 to 0x7F). 0 is reserved.

The actual value of COMMAND in the Value column is described as `[Descriptive_Name]` where "Descriptive_Name" can be found in a 'C' header file which can be obtained on request from Ezurio.

The value of STATUS is similarly defined in a header file which can be obtained from Ezurio by request.

3.1 Information Commands

This group of commands are used to obtain information about the module.

3.1.1 No Operation

This command results in no action other than to convey new FLOW_IN status to the module and get a response packet with the latest status for the FLOW_OUT bits.

It is expected that a host will use this packet to poll for a change in the flow bits.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	4	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_NO_OPERATION]	
3	FLOW_IN	??	Runtime value

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	5	
1	CHANNEL	0	
2	COMMAND	[CMD_NO_OPERATION]	
3	FLOW_OUT	??	Runtime value
4	STATUS	[OK]	Or [INVALID_LICENSE]

3.1.2 Get Connectable, Discoverable, Security Modes

This command is used to get the current mo connectable, discoverable and security modes.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	4	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_GET_MODES]	
3	FLOW_IN	??	Runtime value

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	8	
1	CHANNEL	0	
2	COMMAND	[CMD_GET_MODES]	
3	FLOW_OUT	??	Runtime value
4	STATUS	OK or INVALID_LICENSE	
5	DISCMODE	0..1	1 for discoverable mode
6	CONNMODE	0..7	Bit 0: 1 for connectable mode Bit 1: 1 for Auto Accept Channe1 Bit 2: 1 for Auto Accept Mux
7	SECMODE	0..2	0 = No Auth, No Encryption 1 = Auth, No Encryption 2 = Auth + Encryption

3.1.3 Read Local Bluetooth Address

This command is used to read the Bluetooth address of the module.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	4	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_READ_BLUETOOTH_ADDRESS]	
3	FLOW_IN	??	Runtime value

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	11	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_READ_BLUETOOTH_ADDRESS]	
3	FLOW_OUT	??	Runtime value
4	STATUS	[OK]	
5..10	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address

3.1.4 Information

This command is used to extract information from the module, for example version number.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	5	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_INFORMATION]	
3	FLOW_IN	??	Runtime value
4	INFOTYPE	0..255	

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	14	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_INFORMATION]	
3	FLOW_OUT	??	Runtime value
4	STATUS	[OK]	
5	INFOTYPE	0..255	Echoed from command
6.13	DATA[8]	As per the table below	

The type of information requested is specified by the INFOTYPE parameter, as per the table below.

INFOTYPE = GET_VERSION (0)			
Offset	Field Name	Range	Comments
0	VERSION_SKEW	0..255	Firmware skew. Custom firmware is differentiated with this field
1	VERSION_MAJOR	0..255	Major version number
2	VERSION_MINOR	0..255	Minor version number
3	VERSION_ENG	0..255	Will be 0 for production releases otherwise Engineering release.
4	MAX_INFO_TYPE	N	Max value of INFOTYPE field
5..6	CSR STACK BUILD No	0..65535	Offset 5 = MSB
7	Reserved	Any value	Ezurio private Use

INFOTYPE = GET_MANUFACTURER (1)			
Offset	Field Name	Range	Comments
0..7	Manufacturer Name	E.g. "CSR"	Chip manufacturer, null terminated string

INFOTYPE = GET_CHIP_INFO (2)			
Offset	Field Name	Range	Comments
0..7	Chip Designation	E.g. "BC2-EXT"	Chip designation, null terminated string

INFOTYPE = GET_PHYSICAL_MEDIUM (3)			
Offset	Field Name	Range	Comments
0	Physical Medium	0	0=Bluetooth
1..7	Reserved	0	

INFOTYPE = GET_ RSSI_RANGE_LIMITS (4)			
Offset	Field Name	Range	Comments
0..1	GOLDEN_RECEIVE_RANGE	0..65535	Golden Receive Range
2..3	RECEIVE_LOWER_LIMIT	-32768..32767	Golden Receive Lower limit
4..5	RECEIVE_UPPER_LIMIT	-32768..32767	Golden Receive Upper limit
6..7	Reserved	Any value	Ezurio private Use

INFOTYPE = GET_ ACTUAL_TX_POWER (5)			
Offset	Field Name	Range	Comments
0..1	TX_POWER_IN_DBM	-32768..32767	In dBm. This is a signed 16 bit value. Offset 0 is msb and offset 1 is lsb

INFOTYPE = GET_ MAX_CODEC_OUTPUT_GAIN (6)			
Offset	Field Name	Range	Comments
0..1	Max Output Gain	0..65535	

3.2 Configuration Commands

This group of commands are used to configure the module.

3.2.1 Read 'S' Register

The module is configured using integer values which can be stored in non-volatile memory, similar to the S registers provided in the single point 'AT' module.

Valid register numbers are in the range 0 to 255. Registers 0 to 127 generally specify parameters which can be stored in a 16 bit storage location and registers 128 to 255 can store 32 bit values.

See section 'S Registers' in chapter "Miscellaneous" for a full list of all registers.

The following command is used to read the current value of the S register REGNO.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	5	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_READ_SREG]	
3	FLOW_IN	??	Runtime value
4	REGNO	0 to 255	

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	10	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_READ_SREG]	
3	FLOW_OUT	??	Runtime value
4	STATUS	[OK]	
5	REGNO	0 to 255	Echoed from Command
6..9	REGVAL[]	Register Value	REGVAL[0] is the most significant octet.

3.2.2 Write 'S' Register

This command is used to write a new value to the S register REGNO.

See section 'S Registers' in chapter "Miscellaneous" for a full list of all registers.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	9	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_WRITE_SREG]	
3	FLOW_IN	??	Runtime value
4	REGNO	0 to 255	Registers 250 to 255 inclusive are used for manufacturing/firmware upgrade purpose only. DO NOT WRITE TO THESE REGISTERS
5..8	REGVAL[]	New Register Value	REGVAL[0] is the most significant octet.

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	10	
1	CHANNEL	0	
2	COMMAND	[CMD_WRITE_SREG]	

3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	REGNO	0 to 255	Echoed from Command
6..9	REGVAL[]	Register Value	REGVAL[0] is the most significant octet.

3.2.3 Store 'S' Registers

This command is used to save the current 'S' register value in non-volatile memory so that they survive a power cycle.

COMMAND PACKET Offset	Field	Value	Comments
0	LENGTH	4	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_STORE_SREG]	
3	FLOW_IN	??	Runtime value

RESPONSE PACKET Offset	Field	Value	Comments
0	LENGTH	5	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_STORE_SREG_]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

3.2.4 Default 'S' Registers

This command is used to force all S register values to factory default.

COMMAND PACKET Offset	Field	Value	Comments
0	LENGTH	4	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ DEFAULT_SREG]	
3	FLOW_IN	??	Runtime value

RESPONSE PACKET Offset	Field	Value	Comments
0	LENGTH	5	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ DEFAULT_SREG_]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

3.2.5 Read Communications Parameters

This command is used to read the communications settings as stored in non-volatile storage.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	4	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ READ_COMMSPARM]	
3	FLOW_IN	??	Runtime value

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH		Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ READ_COMMSPARM]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5..8	BAUD[]	1200..921600	Offset 5 is the MSB and offset 9 the LSB
9	STOPBITS	1..2	Stop bits
10	PARITY	0..2	0=None, 1=Odd, 2=Even
11	FLOW	1	1= CTS/RTS handshaking

At the time of writing, only CTS/RTS handshaking is available. It cannot be disabled.

3.2.6 Write Communications Parameters

This command is used to write new communications settings to non-volatile storage.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	11	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_WRITE_COMMSPARM]	
3	FLOW_IN	??	Runtime value
4..7	BAUD[]	1200..921600	Offset 5 is the MSB and offset 9 the LSB
8	STOPBITS	1..2	Stop bits
9	PARITY	0..2	0=None, 1=Odd, 2=Even
10	FLOW	1	1= CTS/RTS handshaking

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	5	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_WRITE_COMMSPARM]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

Note: The communication parameters come into effect after a reset/power cycle.

3.3 Connection Commands

This group of commands are used to establish and break connections.

3.3.1 Set Connectable Mode

This command enables/disables connectable mode and specifies auto accept parameters for channels / muxs.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ CONNECTABLE_MODE]	
3	FLOW_IN	??	Runtime value
4	ENABLE	0..1, 0xFF	0 = Disable, 1=Enable 0xFF = Read current mode
5	ACCEPT	Bit mask	Bit 0: Set to auto accept channel setup Bit 1: Set to auto accept mux setup (CURRENTLY IGNORE – FORCED TO 1 INTERNALLY)

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ CONNECTABLE_MODE]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	CURMODE	0..1	0 = Not connectable 1 = Connectable

In future releases of the firmware, the ability to accept/reject incoming RFCOMM mux setup will be provided. Until then, mux setups will be automatically accepted. The module will use the parameters stored in 'S' Registers 9 and 10 to set the inquiry scan interval and window

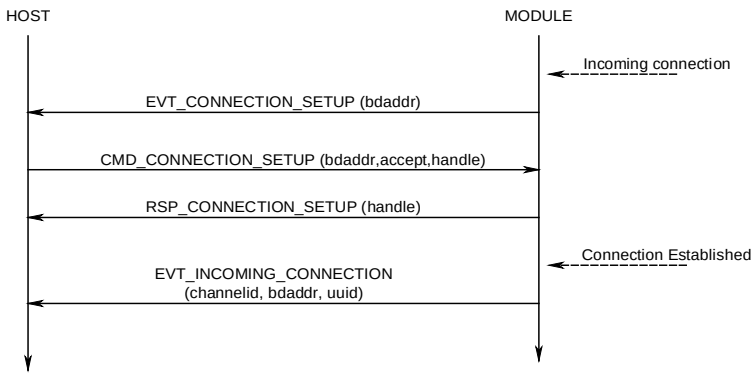
3.3.2 Service Incoming Connection

When the module is in connectable mode, incoming connection requests are passed up to the host via an EVT_CONNECTION_SETUP message. The host accepts or rejects the remote connection request using this message.

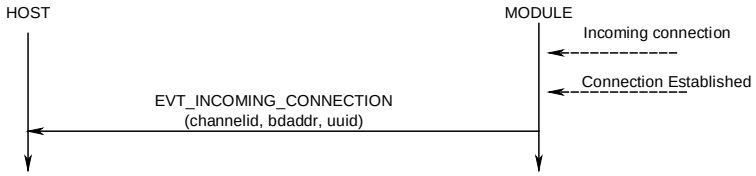
COMMAND PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	12	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ CONNECTION_SETUP]	
3	FLOW_IN	??	Runtime value
4..9	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth addr
10	HANDLE	0..255. Can be any value the host wants to set.	This value is echoed by the module in the response.
11	ACCEPT	0..1	0 = reject, 1 = accept

RESPONSE PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ CONNECTION_SETUP]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	HANDLE	0..255	Echoed from the command

Receipt of the response is **not** an indication that the connection has been established. If the connection is to be accepted, the module will send EVT_INCOMING_CONNECTION when the connection has been fully established, as shown in the message sequence chart below



Note: If auto accept was specified when the module was put into connectable mode, then for incoming connections there will only be an EVT_INCOMING_CONNECTION message.



3.3.3 Make Outgoing Connection

This command is used to make an outgoing connection to a RFCOMM based profile in the remote peer.

Some peers have more than one instance of a given profile. For example, some mobile phones advertise two serial port profiles. The 'make connection' message has an instance field which is used to specify which instance of the remote profile the connection should be made to.

COMMAND PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	13	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ MAKE_CONNECTION]	
3	FLOW_IN	??	Runtime value
4	HANDLE	0..255 Can be any value the host wants to set.	This value is echoed by the module in the response.
5..10	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address
11..12	UUID[]	0x1101 .. 0x11xx	Uuid of the profile to connect to. Offset 11 is the MSB.
13	INSTANCE	0..255	0 is the first instance of the profile, 1 is the second etc.

RESPONSE PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	7	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ MAKE_CONNECTION]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	HANDLE	0..255	Echoed from the command
6	CHANNEL	1..7	Channel Id to be used for subsequent data packets, and also when dropping the connection

If the STATUS field in the response is MPSTATUS_OK, then a connection was successfully established. Any other value is a failure. The timeout specified in 'S' register 16 is used to abort the connection attempt if it takes too long. Likewise, once the connection is established, the value of 'S' register 12 is used to specify the link supervision timeout. Finally the content of 'S' register 11 is used to specify the max frame size to be used by the lower layers – 0 means use default value.

3.3.4 Drop Connection

This command is used to destroy an existing channel.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ DROP_CONNECTION]	
3	FLOW_IN	??	Runtime value
4	HANDLE	0..255 Can be any value the host wants to set.	This value is echoed by the module in the response.
5	CHANNEL	1..7	As was specified in either RSP_MAKE_CONNECTION or EVT_INCOMMING_CONNECTION

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH		Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ DROP_CONNECTION]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	HANDLE	0..255	Echoed from the command

If the STATUS field in the response is MPSTATUS_OK, the request to drop the channel was successfully submitted to the lower stack layers. When the channel is dropped, an EVT_DISCONNECT event is sent to the host.

Note: Prior to v0.1.14, the meaning of this message was:

If the STATUS field in the response is MPSTATUS_OK, then the channel was successfully destroyed. Any other value is a failure.

The meaning of this message was changed post 0.1.14 to make it easier for the host as it now needs to cater for only EVT_DISCONNECT to mean that a connection has been dropped.

3.3.5 Set Modem Lines

Bluetooth Serial Port Profile is capable of exchanging modem signals DTR, DSR, RTS, CTS, DCD and RI over air.

From a host's perspective, it can have DTR, RTS, DCD and RI as output lines. (Note DCD and RI are outputs for modems and 'host' in this context can mean either a PC or a peripheral like a modem).

Additionally UARTs are capable of sending BREAK signals. BREAK output signals are defined as a non-idle state TX pin for a period much greater than the character width at the current baud rate setting.

This command is used to send DTR, RTS, DCD and RI states to the peer device and also to specify a BREAK.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	7	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ CONTROLMODEMLINES]	
3	FLOW_IN	??	Runtime value
4	CHANNEL	1..7	Channel ID of an open channel
5	MODEM	Bit Mask	Bit 0: DTR state Bit 1: RTS state Bit 2: DCD state Bit 3: RI state
6	BREAK	0	For future implementation

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ CONTROLMODEMLINES]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	CHANNEL	1..7	Echoed from command

The STATUS value will be MPSTATUS_OK if the message was successful.

Modem signals sent by the peer device are presented to the host in the message EVT_MODEM_STATUS defined in subsequent chapters.

Note: BREAK signal capability is currently not provided by the lower stack, and so it is mentioned in the context of this command message for future implementation.

3.3.6 SCO Connect

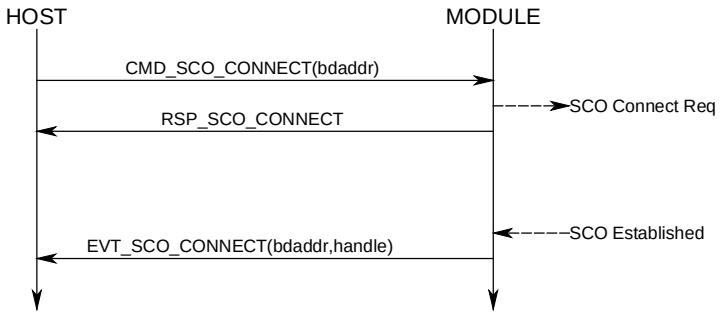
This command is used to establish an SCO channel alongside a pre-existing connection. SCO channels are used to transfer audio between peers and is bi-directional.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	11	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_SCO_CONNECT]	
3	FLOW_IN	??	Runtime value
4..9	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address
10	PKTTYPE	Bit Mask	Bit 0: HV1/EV3 Bit 1: HV2/EV4 Bit 2: HV3/EV5 Bit 3: Enhanced Sco

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_SCO_CONNECT]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

The STATUS value will be MPSTATUS_OK if the request was successfully submitted to the stack.

When the SCO channel is established an EVENT called 'SCO Connect' will be sent to the host which will contain a handle to be used to disconnection requests.



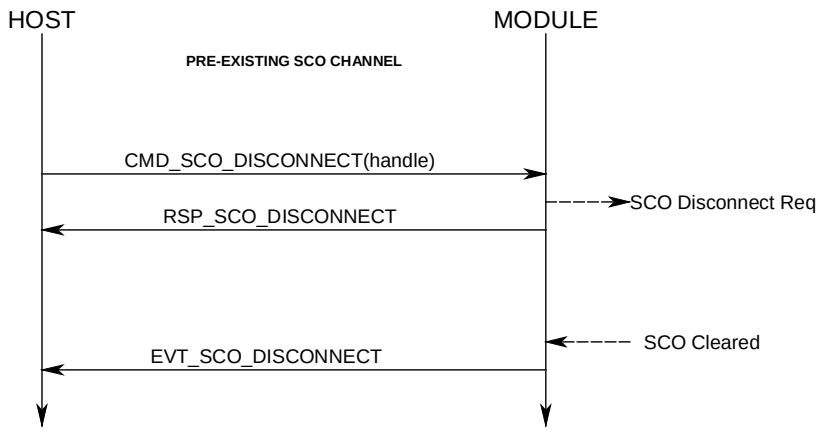
3.3.7 SCO Disconnect

This command is used to disconnect an existing SCO channel identified by a handle which is provided in the SCO establishment event – see Event 'SCO Connect'

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_SCO_DISCONNECT]	
3	FLOW_IN	??	Runtime value
4..5	HANDLE[]	2 byte SCO handle	Handle

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_SCO_DISCONNECT]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

The STATUS value will be MPSTATUS_OK if the message was submitted to the lower stack. When the SCO channel is cleared an EVENT called 'SCO Disconnect' will be sent to the host.



3.3.8 SCO Incoming Setup

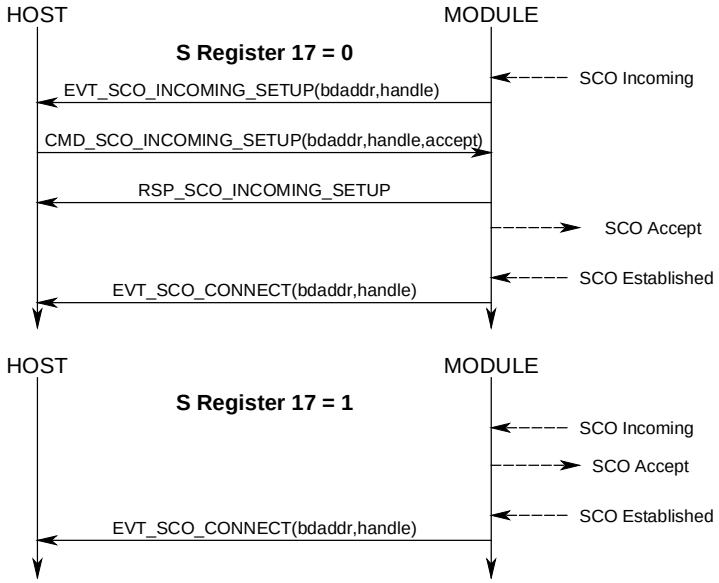
This command is used to accept or reject an incoming SCO channel and is used in response to a 'SCO Incoming Setup' event.

The 'SCO Incoming Setup' event in turn will only be sent to the host if S Reg 17 is 0. By default S Reg 17 is 1 and so incoming SCO channel setup will be automatically accepted.

COMMAND PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	13	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ CONTROLMODEMLINES]	
3	FLOW_IN	??	Runtime value
4..9	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address, echoed from EVT_SCO_INCOMING_SETUP
10..11	HANDLE[]	SCO handle	2 byte handle, echoed from EVT_SCO_INCOMING_SETUP
12	ACCEPT	0, 1 or 2	0 to reject, 1 to accept SCO and 2 to accept eSCO

RESPONSE PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ CONTROLMODEMLINES]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

The BDADDR and HANDLE fields **MUST** be copied from the EVT_SCO_INCOMING_SETUP packet.



3.3.9 Sniff Request

This command is used to enter or exit Sniff mode for a given open connection. If the ATTEMPT value is set to 0, then the device will take that to mean exit sniff mode.

The response packet will be received immediately. The host shall wait for an EVT_LOWPOWER_MODE event for actual notification of success or failure.

COMMAND PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	18	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_SNIFF_REQUEST]	
3	FLOW_IN	??	Runtime value
4..9	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address
10..11	ATTEMPT[]	Sniff Attempt	Millisecs. Offset 10 is MSB
12..13	TIMEOUT[]	Sniff Timeout	Millisecs. Offset 12 is MSB
14..15	MIN_INT[]	Sniff Minimum Interval	Millisecs. Offset 14 is MSB
16..17	MAX_INT[]	Sniff Maximum Interval	Millisecs. Offset 16 is MSB

RESPONSE PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_SNIFF_REQUEST]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

The STATUS value will be MPSTATUS_OK if the request was successfully submitted to the stack.

Minimum value of ATTEMPT is MIN_SNIFF_ATTEMPT_MSEC as defined in the header file BmHostProtocol.H. (12ms at the time of writing this manual).

Minimum value of TIMEOUT is MIN_SNIFF_TIMEOUT_MSEC as defined in the header file BmHostProtocol.H. (12ms at the time of writing this manual).

Minimum value of MIN_INT is MIN_SNIFF_MININTERVAL_MSEC as defined in the header file BmHostProtocol.H. (100ms at the time of writing this manual).

Minimum value of MAX_INT is MIN_SNIFF_MAXINTERVAL_MSEC as defined in the header file BmHostProtocol.H. (100ms at the time of writing this manual).

Maximum value of MAX_INT is MAX_SNIFF_MAXINTERVAL_MSEC as defined in the header file BmHostProtocol.H. (2000ms at the time of writing this manual).

Maximum value of MIN_INT cannot exceed the value specified in MAX_INT.

Maximum values of ATTEMPT and TIMEOUT cannot exceed value of MIN_INT.

3.3.10 Park Request

This command is used to enter or exit PARK mode for a given open connection. If either MIN_INT or MAX_INT value is set to 0, then the device will take that to mean exit park mode.

The response packet will be received immediately. The host shall wait for an EVT_LOWPOWER_MODE event for actual notification of success or failure.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	14	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_PARK_REQUEST]	
3	FLOW_IN	??	Runtime value
4..9	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address
10..11	MIN_INT[]	Park Minimum Interval	Millisecs. Offset 10 is MSB
12..13	MAX_INT[]	Park Maximum Interval	Millisecs. Offset 12 is MSB

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_PARK_REQUEST]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

The STATUS value will be MPSTATUS_OK if the request was successfully submitted to the stack.
Minimum value of MIN_INT is MIN_PARK_MININTERVAL_MSEC as defined in the header file BmHostProtocol.H. (10ms at the time of writing this manual).

Minimum value of MAX_INT is MIN_PARK_MAXINTERVAL_MSEC as defined in the header file BmHostProtocol.H. (10ms at the time of writing this manual).
Maximum value of MAX_INT is MAX_PARK_MAXINTERVAL_MSEC as defined in the header file BmHostProtocol.H. (40000ms at the time of writing this manual).
Maximum value of MIN_INT cannot exceed the value specified in MAX_INT.

3.3.11 RSSI and Link Quality

This command is used to obtain the RSSI and Link Quality values for a given open connection. This is a parameter associated with the ACL connection to a peer device and has no meaning with channel ids.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	10	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_RSSI_LINKQUAL]	
3	FLOW_IN	??	Runtime value
4..9	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth addr

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH		Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_RSSI_LINKQUAL]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5..10	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth addr
11	RSSI	-128 to 127	Rssi value. Will be 0 if the signal is within the golden range
12	LINKQUAL	0 – 255	

The definitions of RSSI and LINKQUAL are paraphrased from the Bluetooth specification as follows:-

3.3.11.1 RSSI

This value is the difference between the measured Received Signal Strength Indication (RSSI) and the limits of the Golden Receive Power Range (see below for definition). Any positive RSSI value returned by the Host Controller indicates how many dB the RSSI is above the upper limit, any negative value indicates how many dB the RSSI is below the lower limit. A value of zero indicates that the RSSI is inside the Golden Receive Power Range. Note: how accurate the dB values will be depends on the Bluetooth hardware. The only requirements for the hardware are that the Bluetooth device is able to tell whether the RSSI is inside, above or below the Golden Device Power Range.

3.3.11.2 GOLDEN RECEIVE POWER RANGE

The lower threshold level of the golden receive power range corresponds to a received power between -56 dBm and 6 dB above the actual sensitivity of the receiver. The upper threshold level is 20 dB above the lower threshold level to an accuracy of +/- 6 dB

3.3.11.3 LINK QUAL

Link_Quality is a value from 0-255, which represents the quality of the link between two Bluetooth devices. The higher the value, the better the link quality is. Each Bluetooth module vendor will determine how to measure the link quality.

In the case of CSR, this value is a measure of BER (Bit Error Rate).

3.3.12 Stream Connect

There is a concept of pipes of data where each pipe is a sink and a source. For example, when an SCO channel is established, it can be a source or sink of data. Likewise, the Bluetooth module has a PCM interface which has 3 slots and each slot can be viewed as a pipe too.

This command is used to connect a 'source' of data to a 'sink' which consumes that data.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	10	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_STREAM_CONNECT]	
3	FLOW_IN	??	Runtime value
4	TYPE	See note 1 below	Source Stream Type
5..6	HANDLE[]	2 byte handle id	Source handle
7	TYPE	See note 1 below	Sink Stream Type
8..9	HANDLE[]	2 byte handle id	Sink handle

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	11	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_STREAM_CONNECT]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	TYPE	See note 1 below	Source Stream Type
6..7	HANDLE[]	2 byte handle id	Source handle
8	TYPE	See note 1 below	Sink Stream Type
9..10	HANDLE[]	2 byte handle id	Sink handle

The STATUS value will be MPSTATUS_OK if the Source was successfully connected to the Sink.

Note 1: Source or Sink Stream Type values are:-
01 - SCO channel and the handle value is as returned in the EVT_SCO_CONNECT message.
02 - PCM channel and valid handle values are 0 to 2 inclusive which map to the slots on the pcm highway.
03 - Reserved for future use.

3.3.13 Stream Disconnect

There is a concept of pipes of data where each pipe is a sink and a source.

This command is used to disconnect a 'sink' which is a consumer of data.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	10	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_STREAM_DISCONNECT]	
3	FLOW_IN	??	Runtime value
4	TYPE	See note 1 below	Source Stream Type
5..6	HANDLE[]	2 byte handle id	Source handle
7	TYPE	See note 1 below	Sink Stream Type
8..9	HANDLE[]	2 byte handle id	Sink handle

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	8	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_STREAM_DISCONNECT]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	TYPE	See note 1 below	Sink Stream Type
6..7	HANDLE[]	2 byte handle id	Sink handle

The STATUS value will be MPSTATUS_OK if the Source was successfully disconnected from the Sink.

3.4 Inquiry Commands

This group of commands are used to performing inquiries and putting the module into discoverable mode.

3.4.1 Inquiry Request

This command is used to perform a Bluetooth inquiry.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	7	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_INQUIRY_REQ]	
3	FLOW_IN	??	Runtime value
4	MAXRESP	1..255	Maximum number of responses before aborting the inquiry procedure
5	TIMEOUT	1..120	Time in seconds, before aborting the inquiry procedure.
6	FLAGS	0	Future use

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	7	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_INQUIRY_REQ]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	TOTAL	??	The total number of inquiry responses that were received from peers.
6	DUMP	??	The total number of inquiry result events that were NOT sent because the transmit buffer of the module was full. This will be as a result of the host deasserting its RTS line.

As a result of this command, as and when peer devices respond with inquiry responses, for each inquiry response, an event EVT_INQUIRY_RESULT is sent to the host.

When the number of inquiry responses specified in the command are received OR the specified time has elapsed, the response will be sent to indicate to the host that the inquiry procedure is complete.

If the DUMP field in the response is non-zero it is indicating that the host is not reading it's receive buffer fast enough and is resulting in RTS being deasserted towards the module.

3.4.2 Set Discoverable Mode

This command is used to enable/disable discoverable mode.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	5	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ DISCOVERABLE_MODE]	
3	FLOW_IN	??	Runtime value
4	ENABLE	0..1, 0xFF	0 = Disable, 1=Enable 0xFF = Read current mode

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ DISCOVERABLE_MODE]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	CURMODE	0..1	0 = Not discoverable 1 = discoverable

The module will use the parameters stored in 'S' Registers 7 and 8 to set the inquiry scan interval and window

3.5 Pairing Commands

This group of commands are used to perform either incoming or outgoing pairings and to manage the trusted device database which resides in the module.

The trusted device database is database with a two tables, each with records of two fields. One field is the Bluetooth address of a paired device and the other is used to store the link key. For security reasons, it is not possible to put the link key on the UART, hence treat it as an invisible field.

One database is classed a ROLLING database and is used to store new pairing information as they happen. If the database is full, then the oldest is discarded to make space for the latest one.

The other database is classed as a PERSISTANT database which stores pairing information which cannot ONLY be deleted when a new pairing is initiated to that device OR on request from the host.

The host protocol provides for a command to transfer a record between these two databases. In addition there is a command for the host to determine if a device is trusted.

Note: Pairing initiation or accept procedures can only be entered if there are no existing connections.

3.5.1 Pair Initiate

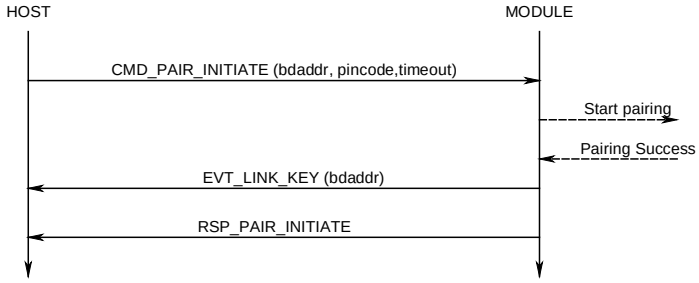
This command is used to initiate a pairing with a peer device which is assumed to be ready and waiting for a pairing.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	28	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ PAIR_INITIATE]	
3	FLOW_IN	??	Runtime value
4	TIMEOUT	5..120	Pairing timeout in seconds
5..10	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth addr of device to be paired
11..27	PIN[]	17 byte string array	Null Terminated Pin code String. Max pin code length is 16.

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	5	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ PAIR_INITIATE]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

If pairing is successful the event message EVT_LINK_KEY will be sent to the host before the response, to close the procedure, as shown in the message sequence chart below.

Note: Pair Initiate will fail if there are any existing open connections. The status byte in the response will have an appropriate value.



3.5.2 Pair Accept

This command is used to put the module into a mode such that it accepts a peer initiated pairing. While in this mode, it is not possible to initiate pairing, inquiry or make connections.

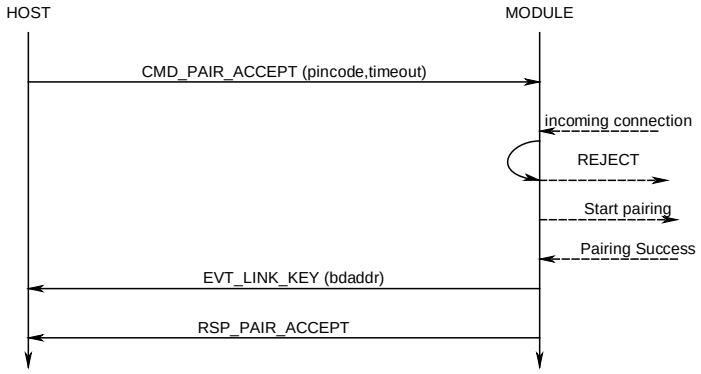
In addition, any incoming connections will be silently rejected.

COMMAND PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	22	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_PAIR_ACCEPT]	
3	FLOW_IN	??	Runtime value
4	TIMEOUT	5..120	Pairing timeout in seconds
5..21	PIN[]	17 byte string array	Null Terminated Pin code String. Max pin code length is 16.

RESPONSE PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	5	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_PAIR_ACCEPT]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

If pairing is successful the event message EVT_LINK_KEY will be sent to the host before the response, to close the procedure, as shown in the message sequence chart below.

Note: Pair Accept will fail if there are any existing open connections. The status byte in the response will have an appropriate value.



3.5.3 PinCode

This command is used to send a pincode in response to an EVT_PINCODE_REQUEST message.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	27	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_PINCODE]	
3	FLOW_IN	??	Runtime value
4..9	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address of device requesting the pairing
10..26	PIN[]	17 byte string array	Null Terminated Pin code String. Max pin code length is 16.

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	5	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_PINCODE]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

If pairing is successful the event message EVT_LINK_KEY will be sent to the host after the response.

3.5.4 Trusted Database Record Count

This command is used to obtain the number of trusted devices in the database specified.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	5	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ TRUSTED_DB_COUNT]	
3	FLOW_IN	??	Runtime value
4	DBTYPE	0..1	0 = ROLLING DATABASE 1 = PERSISTANT DATABASE

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	8	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ TRUSTED_DB_COUNT]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	DBTYPE	0..1	Echoed from command
6	COUNT	0..N	Number of trusted devices in this database
7	MAXCOUNT	0..N	Maximum number of devices that can be stored in this database

Note: ROLLING database is used to store all new pairings. It is up to the host to transfer a record from ROLLING to the PERSISTANT database.

3.5.5 Trusted Database Read Record

This command is used to read the Bluetooth address of the ITEMNO pairing in the database specified, counted from the top. ITEMNO is indexed from 1.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_TRUSTED_DB_READ]	
3	FLOW_IN	??	Runtime value
4	DBTYPE	0..1	0 = ROLLING DATABASE 1 = PERSISTANT DATABASE
5	ITEMNO	1..N	Item number to read

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	13	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_TRUSTED_DB_READ]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	DBTYPE	0..1	Echoed from command
6	ITEMNO	1..N	Echoed from command
7..12	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth addr. Will be all 0's if the item specified does not exist

3.5.6 Trusted Database Delete Record

This command is used to delete a pairing from the databases. It is not necessary to specify the database type, as any type of instance is deleted.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	10	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ TRUSTED_DB_DELETE]	
3	FLOW_IN	??	Runtime value
4..9	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth addr of device to be unpaired

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	11	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ TRUSTED_DB_DELETE]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5..10	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth addr, echoed from the command

3.5.7 Trusted Database Change Type

This command is used to transfer a record to the database specified.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	11	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_TRUSTED_DB_CHANGETYPE]	
3	FLOW_IN	??	Runtime value
4	DBTYPE	0..1	
5..10	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth addr

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	12	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_TRUSTED_DB_CHANGETYPE]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	DBTYPE	0..1	Echoed from command
6..11	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth addr, echoed from the command

3.5.8 Trusted Database Is Peer Trusted

This command is used to check if a device is trusted.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	10	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ TRUSTED_DB_ISTRUSTED]	
3	FLOW_IN	??	Runtime value
4..9	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth addr

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	11	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_ TRUSTED_DB_ISTRUSTED]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5..10	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth addr, echoed from the command

The STATUS value will be MPSTATUS_OK if the device is trusted, any other value means not trusted.

3.6 Miscellaneous Commands

3.6.1 Set/Get Security Mode

This command is used to set or get the current security mode of the module. The module can be in one of three modes. No authentication (0), Authentication Only (1), Authentication and Encryption (2). These modes are applied to all connection attempts, whether inbound or outbound.

Specifying a value of 0xFF means leave the mode as it is, but inform the host with regards to current mode.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	5	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_SECURITY_MODE]	
3	FLOW_IN	??	Runtime value
4	SECMODE	0..2, 0xFF	0 = No Auth, No Encryption 1 = Auth, No Encryption 2 = Auth + Encryption 0xFF = Get current mode

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_SECURITY_MODE]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	SECMODE	0..2	Current mode

3.6.2 Get Remote Friendly Name

This command is used to get the friendly name of the specified peer device.

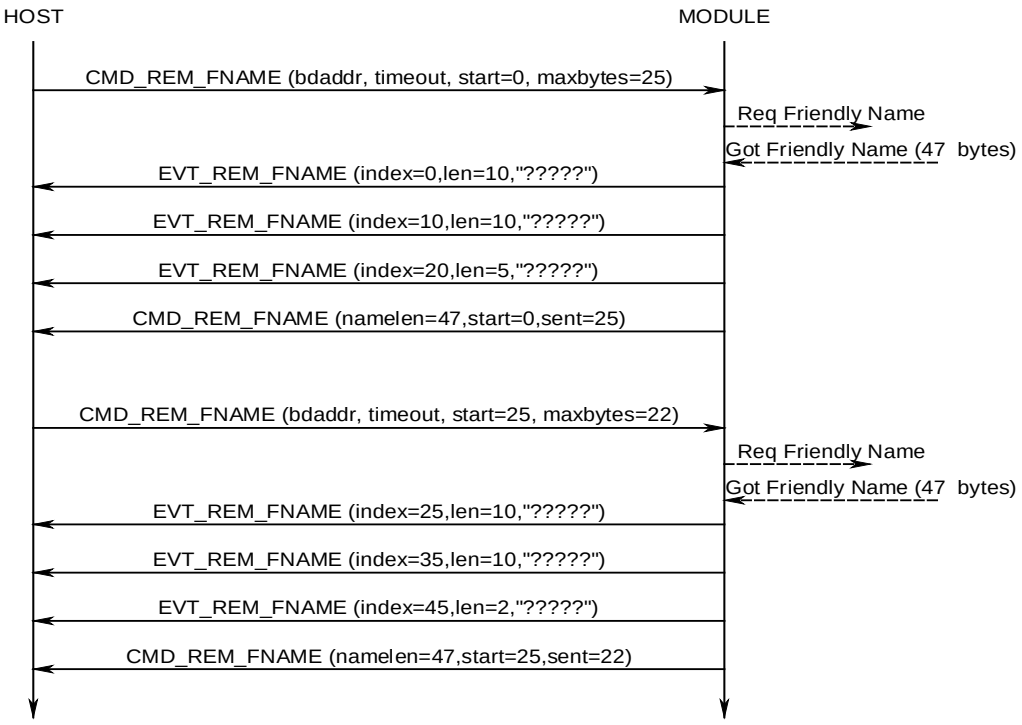
According to the Bluetooth specification a friendly name can be up to 248 bytes long. Sending this name in one go to the host could violate the max packet length capability of the host due to memory restrictions in the host OR transmit buffers in the module may not be able to cope. Therefore, the mechanism for getting the name to the host is via event packets EVT_REM_FNAME. The host decides how many bytes of the name is to be passed up to it via these events from the offset it also specifies. This implies that in a memory constraint environment, it will be possible to relay the name to the host using multiple commands.

For example, if the host has space for only 10 bytes and a peer happens to have a very long name, the host can ask for 10 byte fragments of the name over multiple get name requests.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	13	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_GET_REM_FNAME]	
3	FLOW_IN	??	Runtime value
4..9	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth addr
10	TIMEOUT	1..120	Timeout in seconds
11	START	n	Offset into the friendly name string
12	MAXBYTES	m	Max characters to read

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH	8	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_GET_REM_FNAME]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	NAMELEN	0..248	Actual size of the friendly name
6	START	n	Echoed from the command
7	SENTLEN	s	Total number of bytes sent

Note: SENTLEN could be less than MAXBYTES. It can happen if there is no space in the module's TX buffer to send events.



3.6.3 Get Local Friendly Name

This command is used read the local friendly name which is stored in non volatile memory.

Unlike the remote friendly name where there is no control over the max length, the local friendly name is limited to 31 characters.

This length still may too big to send to the host in one packet. Therefore the name is sent in a similar fashion to 'get friendly name'. However in this case the event EVT_LCL_FNAME is used to get the name to the host.

COMMAND PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_GET_LCL_FNAME]	
3	FLOW_IN	??	Runtime value
4	START	n	Offset into the friendly name string
5	MAXBYTES	m	Max characters to read

RESPONSE PACKET			
Offset	Field	Value	Comments
0	LENGTH		Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_GET_LCL_FNAME]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	NAMELEN	0..31	Actual size of the friendly name
6	START	n	Echoed from the command
7	SENTLEN	s	Total number of bytes sent in preceding events.

The name string is sent to the host in EVT_LCL_FNAME packets. See description of Get Remote Friendly name

Note: SENTLEN could be less than MAXBYTES. It can happen if there is no space in the module's TX buffer to send events.

3.6.4 Set Local Friendly Name

This command is used to set the local friendly name in non-volatile memory so that it is used after a power up/reset cycle.

Since the module can cope with large packets, the name sent to the module in a single command packet as a null terminated string.

COMMAND PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	36	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_SET_LCL_FNAME]	
3	FLOW_IN	??	Runtime value
4	FLAGS	1 MUST ALWAYS BE SET TO 1	Bit 0: Set to store in nonvol memory Bits 1..7: Future use
5	NAMELEN	1..30	
6..36	NAME[31]	Null terminated string	Not more than 30 characters

RESPONSE PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH		Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_SET_LCL_FNAME]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

Note: Future releases will allow setting the friendly name immediately in the baseband and/or storage in non-volatile memory. The mechanism for this will be via bit flags in field FLAGS.

3.6.5 Get Digital/Analog I/O

This command is used read the states of up to 16 digital input lines and optionally request an analogue input reading. This response packet contains 2 octets containing the digital input states. If an analogue input reading is requested then the ADC reading in the range 0..1800 will be supplied in an EVENT_ADC event.

COMMAND PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_GET_IO]	
3	FLOW_IN	??	Runtime value
4	digId	0	0 = Onboard Digital I/o 1 = I/O Mapped as GPIO pins
5	analogId	0..2	0 = No ADC access 1 = ANAL_1 (MAIO_0) 2 = ANAL_2 (MAIO_1)

RESPONSE PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH		Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_GET_IO]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5	digId	0	Echoed from command packet
6..7	digIn[2]	??	Digital inputs 0 to 15

Note: If **`analogId`** field in the command is 1 or 2 and EVENT_ADC will be generated when the ADC is read and available

3.6.6 Set Digital I/O

This command is used to control the states of up to 16 digital output lines. The appropriate gpio pin will have been set to an output pin via S Register 23.

COMMAND PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH	6	Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_SET_IO]	
3	FLOW_IN	??	Runtime value
4	ioId	1	1 = I/O Mapped as GPIO pins
5..6	ioVal[2]	0000..FFFF	

RESPONSE PACKET			
Offset	**Field**	**Value**	**Comments**
0	LENGTH		Fixed
1	CHANNEL	0	Fixed
2	COMMAND	[CMD_SET_IO]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	

4 MODULE EVENTS

This section describes all module originated asynchronous events in detail and is specified via the EVENT field of all event packets.

The description for each event below is in the form of an event packet tables.

Each event has a unique EVENT value in the range 129 to 255 (0x81 to 0xFF), 0x80 is reserved.

The actual value of EVENT in the Value column is described as `[Descriptive_Name]` where "Descriptive_Name" can be found in a 'C' header file which can be obtained on request from Ezurio.

4.1 Inquiry Events

This group of events are inquiry related.

4.1.1 Inquiry Result

This event is used to send the inquiry response from a peer as a result of an inquiry request.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	13	
1	CHANNEL	0	
2	EVENT	[EVT_ INQUIRY_RESULT]	
3	FLOW_OUT	??	Runtime value
4..9	BDADDR[6]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address of responding device
10..12	DEVCLASS[3]	0x000000 .. 0xFFFFFF	Device class of responding device

4.2 Information Events

This group of events are used to convey information about the module, for example to status.

4.2.1 Unknown Command

This event is used to inform the host that a command was received with an unknown COMMAND value. The COMMAND value is echoed in offset 4.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH		
1	CHANNEL		
2	EVENT	[EVT_UNKNOWN_COMMAND]	
3	FLOW_OUT	??	Runtime value
4	command	xx	COMMAND value echoed from command

4.2.2 Status

This event is used to asynchronously send current status to the host. This event is sent to the host after power up to inform the host that the module is ready and operational. The information contained in this message can also be obtained by sending the CMD_GET_MODES command.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	8	
1	CHANNEL	0	
2	EVENT	[EVT_ STATUS]	
3	FLOW_OUT	??	Runtime value
4	STATUS	OK or INVALID_LICENSE	
5	DISCMODE	0..1	1 for discoverable mode
6	CONNMODE	0..7	Bit 0: 1 for connectable mode Bit 1: 1 for Auto Accept Channe1 Bit 2: 1 for Auto Accept Mux
7	SECMODE	0..2	0 = No Auth, No Encryption 1 = Auth, No Encryption 2 = Auth + Encryption

4.2.3 Invalid Packet Size

This event is used to inform the host that a command packet has been received whose length does not .

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	7	
1	CHANNEL	0	
2	EVENT	[EVT_INVALID_PKTSIZE]	
3	FLOW_OUT	??	Runtime value
4	COMMAND	1..127	Echoed from the command
5	ACTSIZE	A	Actual size of the packet
6	DESSIZE	D	Desired size of the packet

4.3 Connection Events

This group of events are connection related.

4.3.1 Connection Setup

This event is used to inform the host that a remote device is requesting a connection. The host shall respond with a CMD_CONNECTION_SETUP with an accept or reject flag.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	12	
1	CHANNEL	0	
2	EVENT	[EVT_ CONNECTION_SETUP]	
3	FLOW_OUT	??	Runtime value
4..9	BDADDR[6]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address of device requesting connection
10..11	UUID[]	Server profile uuid the peer wishes to connect to	0x1101 = SPP 0x1108 = HEADSET 0x1112 = AUDIO GATEWAY 0x1103 = DUN

The UUID field tells the host to which server profile the peer wishes to connect to.

4.3.2 Incoming Connection

This event is used to inform the host that an incoming connection has been established.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	13	
1	CHANNEL	0	
2	EVENT	[EVT_ CONNECTION_SETUP]	
3	FLOW_OUT	??	Runtime value
4	CHANNEL	1..7	Channel ID to be used to send/receive data
5..10	BDADDR[6]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address of device requesting connection
11..12	UUID[]	Server profile uuid the peer wishes to connect to	0x1101 = SPP 0x1108 = HEADSET 0x1112 = AUDIO GATEWAY 0x1103 = DUN

The UUID field tells the host to which server profile the peer has connected to.

4.3.3 Disconnect

This event is used to inform the host that a connection has been dropped by the remote device.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	
1	CHANNEL	0	
2	EVENT	[EVT_ DISCONNECT]	
3	FLOW_OUT	??	Runtime value
4	CHANNEL	1..7	Channel number
5	REASON	0..255	As per the table below

The reason value is specified in the Bluetooth specification and are reproduced here for your convenience as follows, please note that values in the range 0xF0 to 0xFF are custom values defined for this implementation and do not appear in the Bluetooth specification.

0x01	Unknown HCI Command.
0x02	No Connection.
0x03	Hardware Failure.
0x04	Page Timeout.
0x05	Authentication Failure.
0x06	Key Missing.
0x07	Memory Full.
0x08	Connection Timeout.
0x09	Max Number Of Connections.
0x0A	Max Number Of SCO Connections To A Device.
0x0B	ACL connection already exists.
0x0C	Command Disallowed.
0x0D	Host Rejected due to limited resources.
0x0E	Host Rejected due to security reasons.
0x0F	Host Rejected due to remote device is only a personal device.
0x10	Host Timeout.
0x11	Unsupported Feature or Parameter Value.
0x12	Invalid HCI Command Parameters.
0x13	Other End Terminated Connection: User Ended Connection.
0x14	Other End Terminated Connection: Low Resources.
0x15	Other End Terminated Connection: About to Power Off.
0x16	Connection Terminated by Local Host.
0x17	Repeated Attempts.
0xFF	High probability that the remote device went out range for longer than the link supervision timeout or was powered down.

4.3.4 Modem Status

This event is used to convey modem status signals originating from the peer device.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	6	
1	CHANNEL	0	
2	EVENT	[EVT_ MODEM_STATUS]	
3	FLOW_OUT	??	Runtime value

4	CHANNEL	1..7	Channel ID of an open channel
5	MODEMSIG	Bit Mask	Bit 0: DSR state Bit 1: CTS state Bit 2: DCD state Bit 3: RI state
6	BREAKSIG	0	For future implementation

4.3.5 Low Power Mode

This event is used to inform the host that the connection related to the Bluetooth address specified has changed its low power mode to one of ACTIVE, HOLD,SNIFF or PARK. The new mode shall assumed to be the case ONLY if the 'status' field has the value MPSTATUS_OK (== 0).

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	12	
1	CHANNEL	0	
2	EVENT	[EVT_LOW_POWER_MODE]	
3	FLOW_OUT	??	Runtime value
4	STATUS	0 = ok	
5..10	BDADDR[6]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address
11	MODE	New Mode	0 = ACTIVE 1 = HOLD 2 = SNIFF 3 = PARK

4.3.6 SCO Connect

This event is used to inform the host that a SCO connection has been established.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	15	
1	CHANNEL	0	
2	EVENT	[EVT_ SCO_CONNECT]	
3	FLOW_OUT	??	Runtime value
4	STATUS	As appropriate	
5..10	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	
11	COUNT	1..3	Total SCO channels open
12..13	HANDLE[]	2 byte SCO Handle	
14	INCOMING	0..1	1 if SCO was established by peer

4.3.7 SCO Disconnect

This event is used to inform the host that a SCO connection has been cleared.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	15	
1	CHANNEL	0	
2	EVENT	[EVT_SCO_DISCONNECT]	
3	FLOW_OUT	??	Runtime value
12..13	HANDLE[]	2 byte SCO Handle	
14	REASON	XX	

4.3.8 SCO Incoming Setup

This event is used to inform the host that a peer is attempting to establish a SCO connection, and the host needs to accept or reject using the CMD_SCO_INCOMING_SETUP command packet.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	13	
1	CHANNEL	0	
2	EVENT	[EVT_ SCO_CONNECT]	
3	FLOW_OUT	??	Runtime value
4..9	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	
10	COUNT	1..3	Total SCO channels open
11..12	HANDLE[]	2 byte SCO Handle	

When the host sends the CM_SCO_INCOMING_SETUP command packet in response to this event is MUST echo the BDADDR[] and HANDLE[] fields from this event packet.

4.3.9 Remote Features

This event is used to inform the host that the peer just about to connect has the feature list as provided in the message.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	18	
1	CHANNEL	0	
2	EVENT	[EVT_REMOTE_FEATURES]	
3	FLOW_OUT	??	Runtime value
4..9	BDADDR[]	Nap[0,1]:Uap[2]:Lap[3,4,5]	
10..17	FEATURE[]	8 bytes	The feature list as described belo

The feature list as as described below:-

0.0 : 3 Slot packets	4.0 : EV4 packets
0.1 : 5 Slot packets	4.1 : EV5 packets
0.2 : Encryption	4.2 : Reserved
0.3 : Slot Offset	4.3 : AFH capable slave
0.4 : Timing accuracy	4.4 : AFH classification slave
0.5 : Role switch	4.5 : Reserved
0.6 : Hold mode	4.6 : Reserved
0.7 : Sniff Mode	4.7 : 3-slot Enhanced Data Rate ACL packets
1.0 : Park State	5.0 : 5-slot Enhanced Data Rate ACL packets
1.1 : Power control requests	5.1 : Reserved
1.2 : Channel quality driven data rate (CQDDR)	5.2 : Reserved
1.3 : SCO link	5.3 : AFH capable master
1.4 : HV2 packet	5.4 : AFH classification master
1.5 : HV3 packet	5.5 : Enhanced Data Rate eSCO 2 Mbps mode
1.6 : u-law log synchronous data	5.6 : Enhanced Data Rate eSCO 3 Mbps mode
1.7 : A-law log synchronous data	5.7 : Reserved
2.0 : CVSD synchronous data	6.0 : Reserved
2.1 : Paging parameter negotiation	6.1 : Reserved
2.2 : Power control	6.2 : Reserved
2.3 : Transparent synchronous data	6.3 : Reserved
2.4 : Flow control lag (least significant bit)	6.4 : Reserved
2.5 : Flow control lag (middle bit)	6.5 : Reserved
2.6 : Flow control lag (most significant bit)	6.6 : Reserved
2.7 : Broadcast encryption	6.7 : Reserved
3.0 : Reserved	7.0 : Reserved
3.1 : Enhanced Data Rate ACL 2 Mbps mode	7.1 : Reserved
3.2 : Enhanced Data Rate ACL 3 Mbps mode	7.2 : Reserved
3.3 : Enhanced Inquiry Scan	7.3 : Reserved
3.4 : Interlaced Inquiry Scan	7.4 : Reserved
3.5 : Interlaced Page Scan	7.5 : Reserved
3.6 : RSSI with inquiry results	7.6 : Reserved
3.7 : Extended SCO lin (EV3 packets)	7.7 : Extended features

4.4 Miscellaneous Events

4.4.1 Link Key

This event is used to inform the host that a new link key has been created for the device indicated and the result of writing to the ROLLING database.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	11	
1	CHANNEL	0	
2	EVENT	[EVT_ LINK_KEY]	
3	FLOW_OUT	??	Runtime value
4..9	BDADDR[6]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address of paired device
10	DBRESULT	0: Success	Any other value is a failure and the reason is a STATUS value as per MPSTATUS.H

4.4.2 Pin Code Request

This event is used to inform the host that a remote device has requested a pairing and a pin code is required to complete the procedure.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	10	
1	CHANNEL	0	
2	EVENT	[EVT_ PINCODE_REQUEST]	
3	FLOW_OUT	??	Runtime value
4..9	BDADDR[6]	Nap[0,1]:Uap[2]:Lap[3,4,5]	Bluetooth address of pairing device

The host shall send a CMD_PINCODE in response to this event. This event is only received if 'accept pairing while in connectable mode' is enabled via S Register 15.

4.4.3 Local Friendly Name

This event is used to send a fragment of the local friendly name to the host. The maximum length of the fragment is 10, so at least 3 of these events are required to convey a local friendly name, if it has the maximum length of 30.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	16	
1	CHANNEL	0	
2	EVENT	[EVT_LCL_FNAME]	
3	FLOW_OUT	??	Runtime value
4	INDEX	0..29	Start index into the string
5	LEN	1..10	Number of valid characters in the NAME[] field that follows
6..15	NAME[10]	Xx xx xx xx	The name fragment

4.4.4 Remote Friendly Name

This event is used to send a fragment of the remote friendly name to the host. The maximum length of the fragment is 10, so at least 25 of these events are required to convey a remote friendly name, if it has the maximum length of 248.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	16	
1	CHANNEL	0	
2	EVENT	[EVT_REM_FNAME]	
3	FLOW_OUT	??	Runtime value
4	INDEX	0..247	Start index into the string
5	LEN	1..10	Number of valid characters in the NAME[] field that follows
6..15	NAME[10]	Xx xx xx xx	The name fragment

4.4.5 ADC value

This event is used to send the ADC input reading of the ADC channel requested in the most recent CMD_GET_IO command.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	7	
1	CHANNEL	0	
2	EVENT	[EVT_ADC]	
3	FLOW_OUT	??	Runtime value
4	adcId	1..2	ADC channel Id
5	adcValMsb	??	Runtime value
6	adcValMsb	??	Runtime value

4.5 Debug Events

4.5.1 Debug Packet

This event is used to convey debugging information to the host, and will be available in engineering/beta builds only.

EVENT PACKET			
Offset	Field	Value	Comments
0	LENGTH	16	
1	CHANNEL	0	
2	EVENT	[EVT_DEBUG_PACKET]	
3	FLOW_OUT	??	Runtime value
4	TYPE_FLAG	XX	Bit 0: First Packet Bit 1: Last Packet Bit 2..5: Reserved Bit 6..7: Message Type
5..15	DATA[]	Contains Ascii data	String conveying message

5 MISCELLANEOUS

This chapter contains miscellaneous information

5.1 S Registers

This section details all the configuration 'S' registers. Min and Max values are given in decimal, unless the value is prefixed by 0x, in that case the value is in hexadecimal.

RegNo Dec (Hex)	Min	Max	Description
1 (01)	0	0	Host Interface
2 (02)	16	255	Maximum data packet size from module to host
3 (03)	1	0x1F	Server Profile record Mask Bit 0 = SPP Bit 1 = HEADSET Bit 2 = DUN Bit 3 = AUDIO GATEWAY Bit 4 = HANDSFREE
4 (04)	0	1	Default Connectable Mode on power up/reset
5 (05)	0	1	Default Discoverable Mode on power up/reset
6 (06)	0	2	Default Security Mode on power up/reset 0 = No Authentication/No Encryption 1 = Authentication/No Encryption 2 = Authentication/Encryption
7 (07)	10	2550	Inquiry Scan Interval in units of msec, rounded to the nearest 10ms. (10ms will give 11.25)
8 (08)	10	2550	Inquiry Scan Window in units of msec, rounded to the nearest 10ms. (10ms will give 11.25)
9 (09)	10	2550	Page Scan Interval in units of msec, rounded to the nearest 10ms. (10ms will give 11.25)
10 (0A)	10	2550	Page Scan Window in units of msec, rounded to the nearest 10ms. (10ms will give 11.25)
11 (0B)	0	127	Max frame size for lower Bluetooth stack. Recommend this not be changed unless you are absolutely sure what you are doing.
12 (0C)	0	30	Link supervision timeout for connections
13 (0D)	0	1	Auto Accept Mux setup. If this is 1, then incoming RFCOMM channels are automatically accepted.
14 (0E)	0	1	Auto Accept Channel Setup. If this is 1, incoming connections will be automatically accepted. If this is 1, EVT_CONNECTION_SETUP events are not sent to the host when an incoming connection arrives.
15 (0F)	0	1	If this is 1, then pairing requests while in connectable mode are automatically rejected. To accept pairing, the host will need to be in accept

			pairing mode, which is initiated by the CMD_PAIR_ACCEPT host command
16 (10)	2	120	For outgoing connection attempts, this is the timeout before the attempt is abandoned.
17 (11)	0	2	Auto Accept incoming SCO channel. If this is 1, then incoming SCO channels are auto accepted, if set to 2, then eSCO is auto accepted. Set to 0 to get the host to accept/reject the connection.
18 (12)	0	1	Auto Route 1st SCO channel to PCM Slot 0 See Note 1 below
19 (13)	1	4	BootMode. Default is 1 which sets the unit up for a Motorola codec in master mode. A value of 3 sets it up for the OKI7702/5 in master mode
20 (14)	-27	6	Maximum RF transmit power. To read actual power at any moment, read it back via CMD_INFORMATION, subcommand 5
21 (15)	0	X (<=15)	Output gain of Codec A. X is dependent on csr stack build and use CMD_INFORMATION, sub command 06 to get actual maximum value
22 (16)	0	X (<=15)	Output gain of Codec B. X is dependent on csr stack build and use CMD_INFORMATION, sub command 06 to get actual maximum value
23 (17)	0	0x01FF	GPIO Direction register. Setting a bit to 1 will configure the GPIO pin as an output and CMD_SET_IO command packet can be used to control the state.
24 (18)	0	1	Option field for Headset profile service record
25 (19)	0	63	Option field for Handsfree profile service record
26 (1A)	0	1	If set to 1, master/slave role switch requests from a peer will be denied.
27 (1B)	0	1	Some events sent to the host have to be enabled by setting appropriate bits in the register. Bit 0 Enable REMOTE_FEATURES event
126 (7E)	0	0xFFFF	Primer for changing to AT mode. See document "blu2i - AT and Multipoint Mode Change.pdf" for more details
127 (7F)	0	0xFFFF	Change to AT mode using value 0x0100.
128 (80)	0	0xFFFFFF	Module's Class of Device

Notes: (1) SReg 18 is read by the firmware at reset. Hence after setting this register a reset is required for it to be effective. This means the S Register set MUST be committed to non-volatile memory before initiating a reset. The S Registers are stored to non-volatile memory using the command **[CMD_STORE_SREG].**

5.2 COMMAND & EVENT values

The following is a listing of a snapshot of the file BMHOSTPROTOCOL.H at the time of writing this document. Ezurio does NOT guarantee that this listing will be kept up to date.

For development purposes, please send a request to multipoint@Ezurio.com for the latest version of the 'C' header file.

```
//#############################################################################
//The following are COMMAND (octet 2) values in command/response packets
//#############################################################################
#define CMD_NO_OPERATION                                0x01
#define CMD_READ_BDADDR                                 0x02
#define CMD_READ_SREG                                   0x03
#define CMD_WRITE_SREG                                  0x04
. . . .
. . . .
. . . .
//#############################################################################
//The following are EVENT (octet 2) values in event packets
//#############################################################################
#define EVT_STATUS                                      0x81
#define EVT_INVALID_PKTSIZE                             0x82
#define EVT_UNKNOWN_COMMAND                             0x83
#define EVT_INQUIRY_RESULT                              0x84
#define EVT_MODEM_STATUS                                0x85
. . . .
. . . .
. . . .
SEND REQUEST TO multipoint@Ezurio.com FOR LATEST VERSION OF THIS HEADER FILE
```

5.3 STATUS values

The following is a listing of a snapshot of the file MPSTATUS.H at the time of writing this document. Ezurio does NOT guarantee that this listing will be kept up to date.

For development purposes, please send a request to serial@Ezurio.com for the latest version of the 'C' header file.

```
#define MPSTATUS_OK                                     0x00

#define MPSTATUS_ILLEGAL_COMMAND                        0x01
#define MPSTATUS_NO_CONNECTION                          0x02
#define MPSTATUS_HARDWARE_FAIL                          0x03
#define MPSTATUS_PAGE_TIMEOUT                           0x04
. . . .
. . . .
. . . .

SEND REQUEST TO multipoint@Ezurio.com FOR LATEST VERSION OF THIS HEADER FILE
```